Marked Copy Number:

D&K, DW exam.

9 Nov. 2016

*Answer **Ex. 1 on the exam sheet.***

Exam is anonymized: report the number printed on your main exam sheet on all others.
No document authorized for Ex.1. You will have to return this sheet after ≃10mins.

Exercise 1 (Definitions.) **(2+1.5+1 = 4.5 pts)**

1. What is a data Warehouse? (define briefly, introducing the essential caracteristics without explaning further. For instance, just recalling Bill Inmon's definition will be accepted as an answer to this question)

   .....

   .....

   Correction: *cf bill Inmon's def.*

2. What is *Hash partitioning*? What benefits does it bring? Will it help to optimize the evaluation of range-based selection; i.e., a query like `SELECT * FROM employees WHERE salary>1499.99 AND salary <2100`?

   .....

   .....

   .....

   .....

   Correction: *Hash partitioning is a particular kind of horizontal partitioning. It uses a hash function to partition records of a table. This guarantees load balancing among partitions provided the partitioning key is almost unique. It can be used for partition pruning in equality predicates, or for partition-wise joins. Not useful for ranges because values the optimizer cannot identify which partitions contain the data for predicates ">" and "<".*

3. Explain why partitioning table `Sales(time_id, customer_id, product_id, amount)` might prevent the execution of the instruction `UPDATE Sales SET time_id = 10 WHERE time_id=9 AND product_id=4` (mention 2 possible causes in Oracle).

   .....

   .....

   .....

   ..... **Correction:** *1) There may be no partition that can receive a sale with tim_id=10, in the case of RANGE or LIST partitioning on `time_id`. 2) If the table is partitioned on on `time_id`, the update may cause the row to move into another partition. In that case, if the ROW MOVEMENT parameter has not been set to "enabled" the update will fail.*

Marked Copy Number:

D&K, DW exam.

9 Nov. 2016

*Answer **Ex. 2 on the exam sheet, Ex. 3 on separate sheet.***

Exam is anonymized: report the number printed on your main exam sheet on all others.

Exercise 2 (Indexes, views) **(2+1.5+2+1.5+1.5 = 8.5 pts)**

The 2 tables below represent parts of a database recording the inventory of a music store. PK indicates a primary key, and → a foreign key. For questions 1 and 2 you will assume that there are no additional records beyond those displayed on the figure whereas for questions 3, 4 and 5 the figures are a sample of existing records. Note that some records might include `NULLs`.

Inventory

Work_id (PK)	Price	C→ Composer.Id	S→ Style.Id	ISBN	Binding
1	24.0	1	1	1	Case
2	23.0	1	1	1	Comb
3	12.4	4	2	2	Comb
4	24.0	4	2	2	Comb
5	10.0	4	1	3	Crisscross
6	39.99	1	1	4	Stitched
7	20.0	4	1	4	Comb
8	59.99	1	1	5	Case
9	30.0	2	2	6	Case
10	8.99	2	2	7	Case

Style

Id (PK)	Name
1	Vocal
2	Piano

Composer

Id (PK)	Name
1	Rameau
2	Glück
3	Sibelius
4	Dvořák

1. Represent a bitmap Join index that records the composer of each work in the inventory.

   .....

2. Apply the run-length encoding viewed in the lecture to the bitvector that begins with a "1".

   .....

3. For each of the following kind of queries, where the only parameters that may vary are the ones **underlined**, indicate which index or combination of indexes should be built (i.e., which index helps best optimize the query at a reasonable cost - we are assuming that indexes speedup reads even on a table with only thousands of records). You will assume the following cardinality for each attribute: `Work_id` has 10^7 distinct values, `Composer.name` has 10^3, `Styles.name` has 20, `Price` has 10^4, `Binding` has 10, and `ISBN`: 3×10^6:

(a) Q1:

```
SELECT SUM(Price) FROM Inventory I, Style St WHERE I.S=St.Id AND (Name = x OR St.Name = y) ;
```

.....

Correction: *Bitmap join of style on inventory.*

(b) Q2:

```
SELECT COUNT(*) FROM Inventory WHERE Price > x And Price < y ;
```

.....

Correction: *B+ tree on Inventory.Price.*

(c) Q3:

```
SELECT Name FROM Composer WHERE Id=x ;
```

.....

Correction: *B+ tree on Composer.Id.*

(d) Q4:

```
SELECT Work_id FROM Inventory WHERE S=x AND Binding<>y ;
```

.....

Correction: *Bitmap indexes on Inventory.S and Inventory.Binding*

(e) Q5 (Bonus). *[Think carefully. N.B.:* `UPPER` *converts a string to uppercase]*:

```
SELECT MAX(Price) FROM Inventory WHERE ISBN=x AND UPPER(Binding)=y ;
```

.....

Correction: *B+tree in Inventory on (x, Upper(Binding), Price).*

4. Give the SQL instruction to create a materialized view `MyView(Binding, Composer, Nb, Worth)` using at least one `ROLLUP` statement [1]. The view should record, for the two groups `(Binding, composer)` and `(Binding)`, the number of works in the group and the total worth of the group (cf picture).

.....

.....

.....

.....

.....

```
> SELECT * FROM MyView;

Binding Composer Nb  Worth
------- -------- -- ------
Case    Glück     2  38.99
...
Case               4 398.99
...
Stitched          75 150.33
...
```

5. Could `MyView` be exploited to rewrite query below? If yes, give the rewriting. If not, explain what should be added to the view to allow the rewriting then give the rewriting.

```
SELECT Binding, AVG(Price)
FROM Inventory
GROUP BY Binding
```

.....

.....

.....

.....

.....

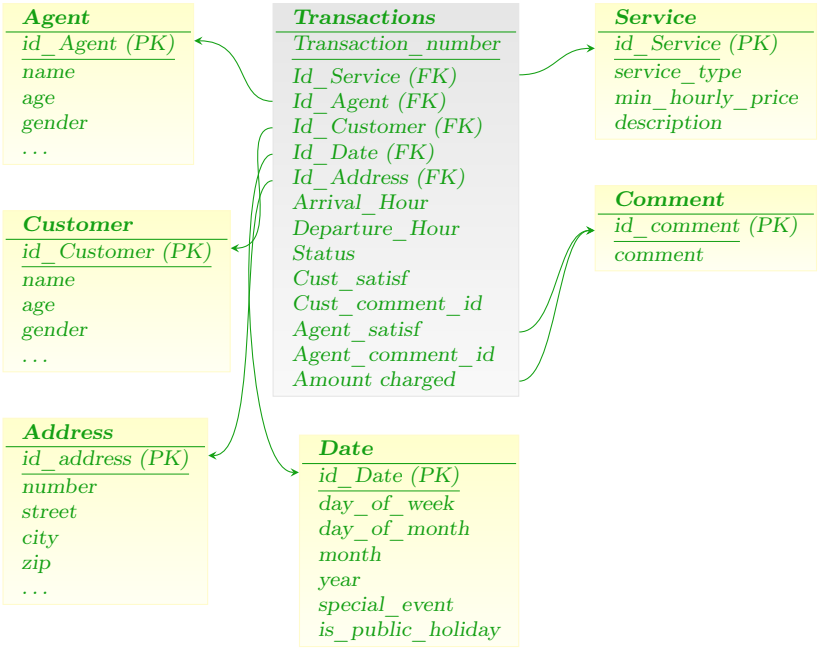
.....

Exercise 3 (Modeling) (2.5+2+2.5 = 7 pts)

Consider a datawarehouse for a company proposing house-cleaning services. We suppose services are described by: Transaction number, Date, Address (full address: street, city, zip), Type of service, Arrival hour, Departure hour, Amount charged, Customer information, Agent information (we will assume a transaction involves a single employee, service, etc.), as well as customer and agent satisfaction grades (integer ≤ 5) and comments. The customer and agent information includes names, age, gender, and other relevant details. We also record for each transaction a status which can only take one the following 3 values `Cancelled`, `Executed`, or `Scheduled` (there is no additional information regarding the status). Each type of service has a minimal hourly price, and a text description (± 10 lines).

1. Propose a star schema for this datawarehouse.

[1] If you don't manage to use ROLLUP you can use `GROUPING SETS` instead but I will remove .5 point



Correction:

2. Identify the fact table, the dimensions (is there a degenerate one?), and measures.

3. Assume the departure hour may have to be corrected (ex: some employee typed an incorrect hour), and the minimal price is always correct but may evolve from one month to the other. How would you adapt your original schema to handle these assumptions? Illustrate your choice on a small database instance (you may restrict the instance to tables affected by the update).